ViewSet

1.list()🡺To get all resources/records [get]

2.retrieve()🡺To get a particular resource [get]

3.create()🡺To create a new resource[post]

4.update()🡺To perform complete updation [put]

5.partial\_update()🡺To perform partial updation [patch]

6.destroy()🡺To delete a resource [delete()]

**When viewset is a best choice?**

**1.If we want to develop a simple CRUD interface**

**2.To develop API very quickly**

**3.To perform standard operations**

**4.If we are not performing any complex operations**

**Note:Routers will map views to urls automatically**

**urls.py**

**from django.urls import path,include**

**from myApp.views import \***

**from rest\_framework.routers import DefaultRouter**

**router=DefaultRouter()**

**router.register(‘endpointname’,TestViewSet,’endpointname’)**

**urlpatterns=[**

**….**

**path(‘api/ ‘,include(router.urls))**

**end user can send request upto this point /api**

**views.py**

**from django.shortcuts import render**

**from rest\_framework.response import Response**

**from myApp.serializers import NameSerializer**

**from rest\_framework.viewsets import ViewSet**

**class TestViewSet(ViewSet):**

**def list(self,request):**

**colors=['red','blue','black','green']**

**return Response({'msg':'Happy bday','colors':colors})**

**def create(self,request):**

**serializer=NameSerializer(data=request.data)**

**#data by browsable api interface**

**if serializer.is\_valid():**

**name=serializer.data.get('name')**

**msg="Hello {} happy bday -:".format(name)**

**return Response({'msg':msg})**

**else:**

**return Response(serializer.errors,status=404)**

**def retrieve(self,request,pk=None):**

**return Response({'msg':'Response through retrieve'})**

**def update(self,request,pk=None):**

**return Response({'msg':'Response through update'})**

**def partial\_delete(self,request,pk=None):**

**return Response({'msg':'Response through partial delete'})**

**def destroy(self,request,pk=None):**

**return Response({'msg':'Response through destroy'})**

**urls.py**

**from django.contrib import admin**

**from django.urls import path,include**

**from myApp.views import TestViewSet**

**from rest\_framework.routers import DefaultRouter**

**router=DefaultRouter()**

**router.register('test\_view\_set',TestViewSet,"test\_view\_set")**

**urlpatterns = [**

**path('admin/', admin.site.urls),**

**path('api/',include(router.urls))**

**]**

**serializers.py**

**from rest\_framework import serializers**

**class NameSerializer(serializers.Serializer):**

**name=serializers.CharField(max\_length=30)**

**INSTALLED\_APPS = [**

**….**

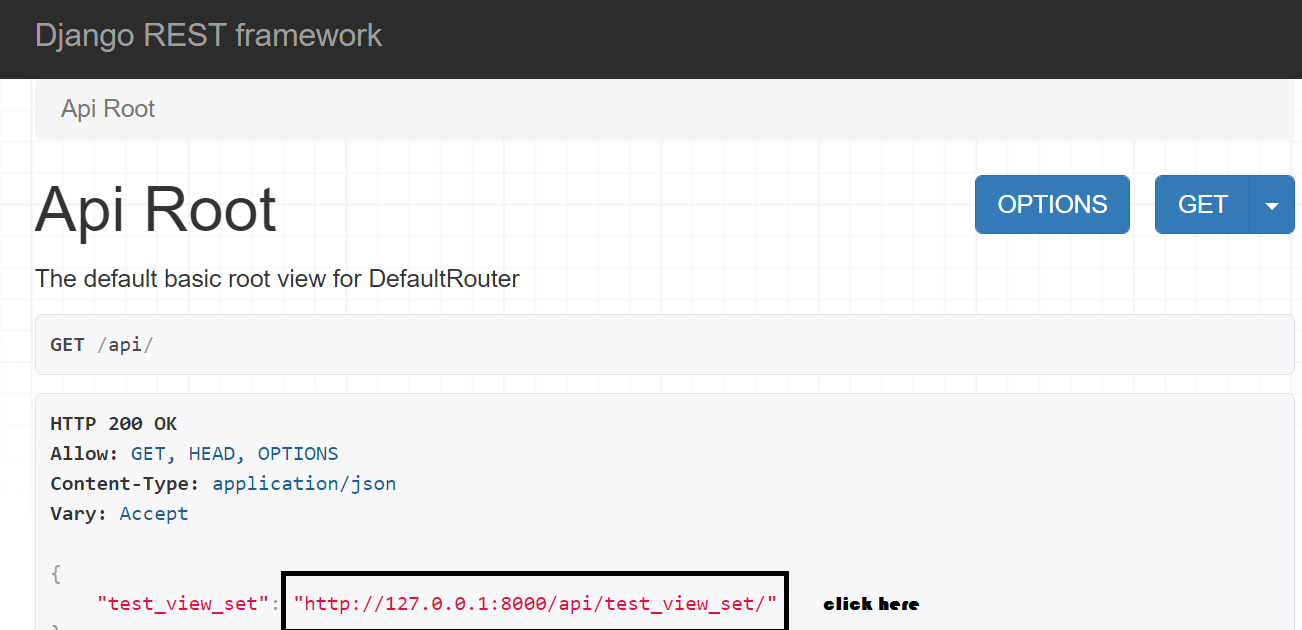
**'myApp',**

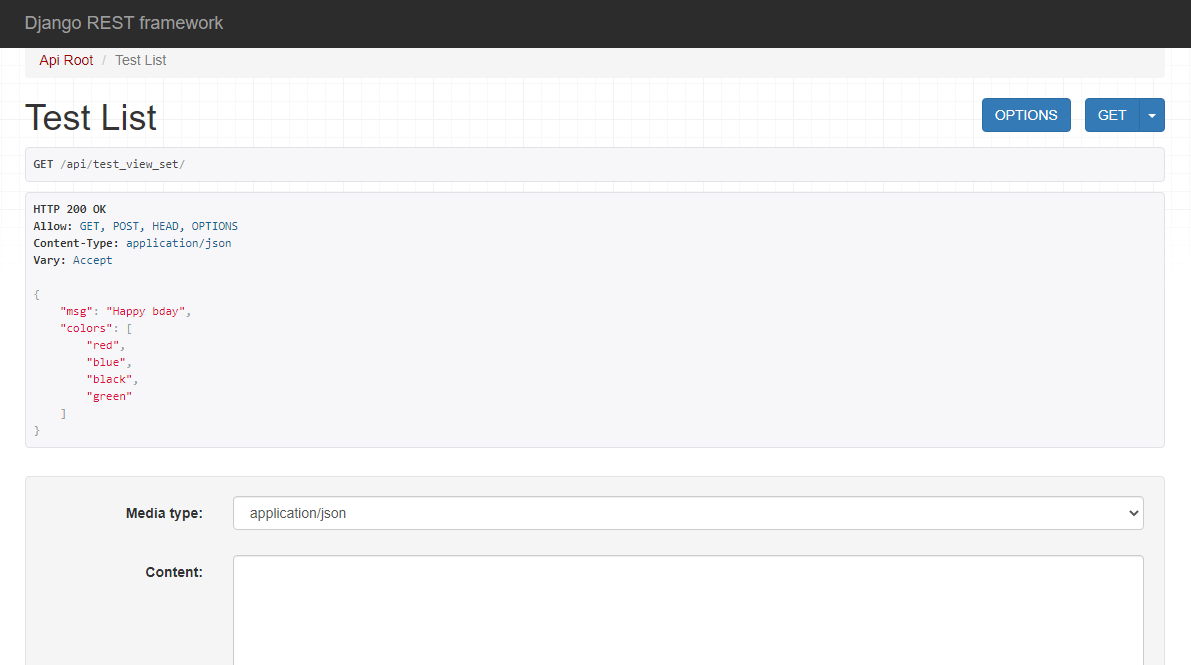
**'rest\_framework'**

**]**

**Runserver and send request**

[**http://127.0.0.1:8000/api/**](http://127.0.0.1:8000/api/)

****

****

|  |  |
| --- | --- |
| **APIView** | **ViewSet** |
| **Present in rest\_framework.views** | **Present in rest\_framework.viewsets** |
| **Method names reflect HTTP methods like,get(),put(),post(),delete()** | **Reflect database model class operations like list(),create(),etc** |
| **We have to map views to urls explicitly** | **DefaultRouters take care of url mapping** |
| **Most of the business logic should be written explicitly** | **Most of the business logic are written automatically** |
| **Length of the code is more** | **Length of the code is less** |
| **API development time more** | **API development time less** |
| **Developer has complete control over logic** | **Developer doesn’t have complete control over logic** |
| **Clear execution flow is possible** | **Clear execution flow is not possible** |
| **Best suited for complex operations like using multiple data sources simultaneously and calling other apis** | **Best suited for developing simple apislike CRUD interface for database models** |

**Demo Application for APIView**

**models.py**

**from django.db import models**

**class Employee(models.Model):**

**eno=models.IntegerField()**

**ename=models.CharField(max\_length=10)**

**esal=models.FloatField()**

**eaddr=models.CharField(max\_length=100)**

**def \_\_str\_\_(self):**

**return self.ename**

**admin.py**

**from django.contrib import admin**

**from myApp.models import Employee**

**class EmployeeAdmin(admin.ModelAdmin):**

**l=['eno','ename','esal','eaddr']**

**admin.site.register(Employee,EmployeeAdmin)**

**myApp/serializers.py**

**from rest\_framework.serializers import ModelSerializer**

**from myApp.models import Employee**

**class EmployeeSerializer(ModelSerializer):**

**class Meta:**

**model=Employee**

**fields="\_\_all\_\_"**

**views.py**

**from django.shortcuts import render**

**from myApp.models import Employee**

**from myApp.serializers import EmployeeSerializer**

**from rest\_framework.views import APIView**

**from rest\_framework.response import Response**

**class EmployeeListApiView(APIView):**

**def get(self,request):**

**qs=Employee.objects.all()**

**serializer=EmployeeSerializer(qs,many=True)**

**return Response(serializer.data)**

**urls.py**

from django.contrib import admin

from django.urls import path

from myApp.views import EmployeeListApiView

urlpatterns = [

path('admin/', admin.site.urls),

path('api/',EmployeeListApiView.as\_view())

]

Perform make migrations,migrate,create super user and runserver

To list the records,we use ListAPIView available in rest\_framework.generics module

views.py

**from django.shortcuts import render**

**from myApp.models import Employee**

**from myApp.serializers import EmployeeSerializer**

**from rest\_framework.generics import ListAPIView**

**from rest\_framework.response import Response**

**class EmployeeListApiView(ListAPIView):**

**queryset=Employee.objects.all()**

**serializer\_class=EmployeeSerializer**

**send request** [**http://127.0.0.1:8000/api/**](http://127.0.0.1:8000/api/)

**To implement search options,**

**Send request along with the query string**

[**http://127.0.0.1:8000/api/?ename=shreenath**](http://127.0.0.1:8000/api/?ename=shreenath)

**o/p🡺all records**

**views.py**

**from django.shortcuts import render**

**from myApp.models import Employee**

**from myApp.serializers import EmployeeSerializer**

**from rest\_framework.generics import ListAPIView**

**from rest\_framework.response import Response**

**class EmployeeListApiView(ListAPIView):**

**queryset=Employee.objects.all()**

**serializer\_class=EmployeeSerializer**

**def get\_queryset(self):**

**qs=Employee.objects.all()**

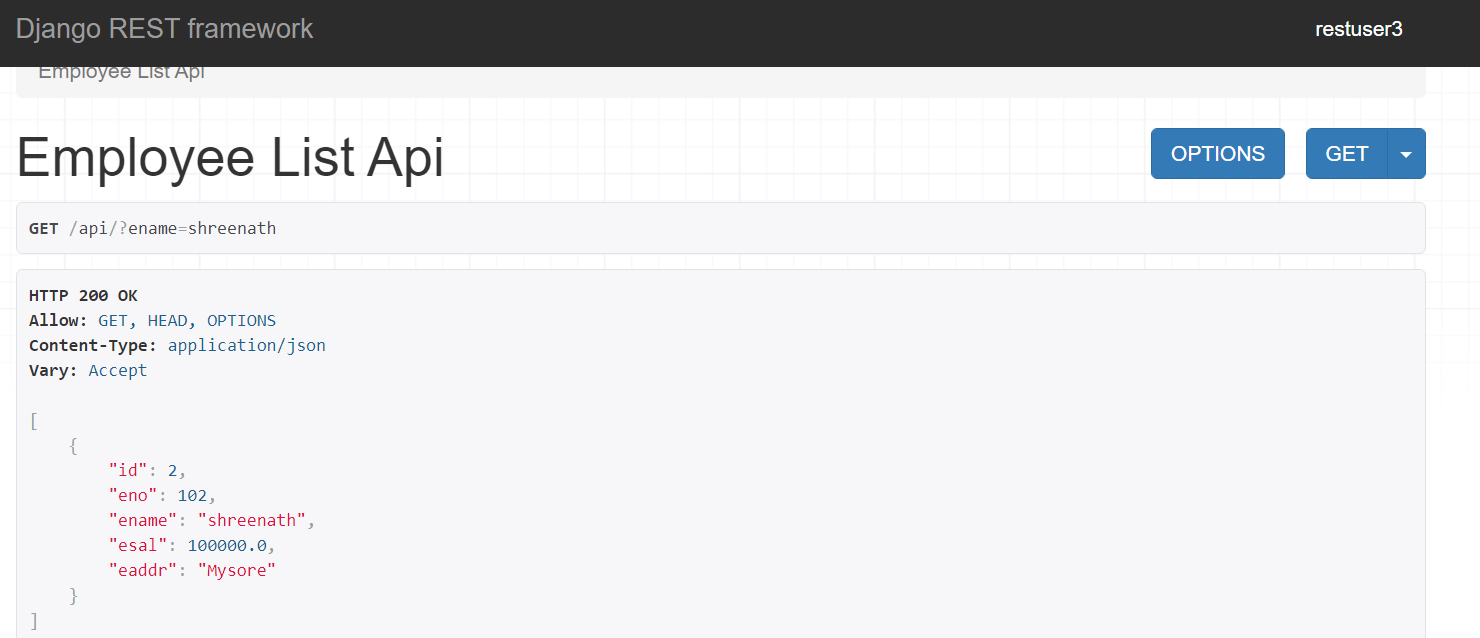
**name=self.request.GET.get('ename')**

**if name is not None:**

**qs=qs.filter(ename\_\_contains=name)**

**return qs**

**send request** [**http://127.0.0.1:8000/api/?ename=shreenath**](http://127.0.0.1:8000/api/?ename=shreenath)

****